The opinion in support of the decision being entered today is *not* binding precedent of the Board.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte NATHAN ABRAMSON and JEFFREY J. VROOM

Appeal 2007-1708
Application 09/209,015
Technology Center 2100

Decided: July 31, 2007

Before FRED E. McKELVEY, *Senior Administrative Patent Judges* ALLEN R. MacDONALD, and ROBERT E. NAPPI, *Administrative Patent Judges.*

NAPPI, *Administrative Patent Judge.*

DECISION ON APPEAL

This is a decision on appeal under 35 U.S.C. § 6(b) of the Final Rejection of claims 1 through 4 and 9 through 17. Claims 5 through 8 have been canceled. For the reasons stated *infra,* we affirm in part the Examiner's rejection of these claims and enter a new ground of rejection.

INVENTION

The invention is directed to a method to automatically map hypertext input fields in a document to software components. The invention useful in dynamic web applications where users input data to a server through a web page, the data can then be used by other software on the server. See pages 1 through 3 of Appellants' Specification. Claim 1 is reproduced below:

1. A method comprising:

rendering a hypertext document including emitting program code for mapping input field names in the hypertext document to software component properties when the hypertext document is rendered, a software component being a server-based component that uses data from the input field for processing;

providing the rendered hypertext document to a user;

receiving from the user input field data entered in a named input field;

determining from the mapping a software component property mapped to the named input field; and

calling the software component for processing the input field data, the mapping being done such that the software component can process the input field data regardless of the spelling of the name of the input field in the hypertext document.

REFERENCE

The reference relied upon by the Examiner is:

Ferris (hereinafter Apple)[1] WO 98/44694 Oct. 8, 1998

EXAMINER'S REJECTIONS

Claims 1 through 3, 9, 12 through 13, and 16 stand rejected under 35 U.S.C. § 102(b) as anticipated by Apple. The Examiner's rejection is set forth on pages 4 through 9 of the Final Office action dated April 19, 2002.

Claims 10 and 14 stand rejected under 35 U.S.C. § 102(b) as anticipated by Apple or in the alternative under 35 U.S.C. § 103(a) as unpatentable over Apple. The Examiner's rejection is set forth on pages 10 and 11of the Final Office action dated April 19, 2002.

Claims 4, 11, 15, and 17 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Apple. The Examiner's rejection is set forth on pages 11 through 13 of the Final Office action dated April 19, 2002.

Throughout the opinion we make reference to the Brief and Reply Brief (filed February 12, 2003 and April 5, 2004 respectively), and the Answer (mailed February 24, 2004) for the respective details thereof.

Examiner's Rejection of Claims 1, 12, and 16.

On page 4 of the Brief, Appellants argue that Apple which is concerned with synchronization of states for recognition of user actions

[1] The Applicant for this PCT document was Apple computer Inc. and the Examiner refers to this document as Apple throughout prosecution. To avoid confusion we similarly refer to the document as "Apple."

differs from the claimed invention which recites mapping input field names in hypertext documents to software components. Further, Appellants argue that the claimed mapping is done regardless of spelling of the name of the input field, where as in Apple "[i]f the source is misspelled or incorrectly provided, the browser will not be able to locate the applet code." (Brief 4.)

In response the Examiner states:

> It is submitted that the claim only calls for any "processing" without explicitly requiring what this processing must be. Therefore, the processing shown by Apple clearly meets the claimed invention. Furthermore, the limitation of "mapping is done so that the software component can process the input field data regardless of the spelling of the name input field in the hypertext document1" in Claim 1 does require that the processing be performed in case the misspelling occurs. As previously noted on page 14 of the previous Office Action (Paper#6), the term "regardless of the spelling" can be, for example, that Apple's "name" on Table One be spelled any way the user desires. That is, the "INPUTFIELD" could be misspelled by the user as, for example, "INPTFLD", and still have operated as described in the Apple's exemplary embodiment.

(Answer 4.)

We are not convinced that the Examiner erred. Claim 1 recites "program code for mapping input field names in the hypertext document to software component properties when the hypertext document is rendered, a software component being a server-based component that uses data from the input field for processing." Thus, the scope of claim 1 includes that there is software for mapping input field names in a hypertext document to a software component property.

Further, claim 1 recites determining a mapping between the software component and the named input field and "calling the software component

for processing the input field data, the mapping being done such that the software component can process the input field data regardless of the spelling of the name of the input field in the hypertext document." We may rely on Appellants' disclosure to determine the meaning of the terms used in the claims. During examination, a claim is given its broadest reasonable construction consistent with the Specification. *In re Prater*, 415 F.2d 1393, 1404-05, 162 USPQ 541, 550-51 (CCPA 1969). "[I]nterpreting what is *meant* by a word *in* a claim 'is not to be confused with adding an extraneous limitation appearing in the specification, which is improper.'" *In re Cruciferous Sprout Litigation,* 301 F.3d 1343, 1348, 64 USPQ2d 1202, 1205 (Fed. Cir. 2002). We find no limitation in claim 1 describing the nature of the processing and to do so would require importing limitations from Appellants' Specification. Similarly, we do not find a claim limitation which recites that the mapping will be done if the name of the input field is misspelled or incorrectly provided (i.e. mapping irrespective of whether the spelling of the name and software component are the same). Claim 1, recites "the mapping being done . . . regardless of the spelling of the name" which we interpret to mean that any spelling will suffice (i.e. any sequence of characters is sufficient to name the input field). However, we do not interpret this limitation as asserted by Appellants, to mean that the mapping will be done if the name is misspelled or incorrectly provided. In other words, we do not interpret claim 1 as limited to performing mapping even if there is a mismatch in spelling between the input field name of the hypertext document and the software component.

Appellants' arguments, on page 4 of the Brief and page 2 of the Reply Brief, which focus on Table 2 on page 16 of Apple have not convinced us of error in the Examiner's rejection. Appellants argue that, in Apple if the values of the code are misspelled or incorrectly provided, the browser will not be able to locate the applet code. As discussed above, we do not find that claim 1 is so limited. Further, in rejecting claim 1, the Examiner relied upon table 1 on page 15 of Apple, which depicts a HTML template that provides a map between entry and object tags. We find no teaching in Apple that the spelling of the name of web objects is in any way limited i.e. there is no restriction that the web objects be named "INPUTFIELD," "BUTTON," or "OUTPUTFIELD."

Further, we are not persuaded by Appellants' argument that claimed invention which recites mapping input field names in hypertext documents to software components, differs from Apple which is concerned with synchronization of states for recognition of user actions. Apple teaches a method of synchronizing data between a client and server where the data is mapped to an object oriented environment used on the server. Thus there is an application on the server which is receiving the input data from the client, i.e. there is a server based component that uses the data input. Further, as discussed above, claim 1 does not recite any limitation regarding the processing that occurs at the server. Thus, we see no claimed distinction between (1) Apple, which is concerned with synchronization of states for recognition of user actions, and (2) the claimed invention, which recites mapping input field names in hypertext documents to software components.

Appellants have not provided separate arguments directed to independent claims 1, 12, and 16. Accordingly, we group these claims together. For the forgoing reasons we sustain the Examiner's rejection of claims 1, 12, and 16.

Examiner's Rejection of Claims 2 through 4.

Appellants argue, on page 5 of the Brief, that claims 2 and 3 are separately patentable as they pertain to using hypertext form tags, whereas Apple teaches using applets instead of HTML forms.

The Examiner states that "[a]s well known, the [*sic,* an] applet can be utilized to provide a hypertext input form as well as the HTML FORM tag element can be used to create the input form." (Answer 5.)

The Examiner may well be correct, but on this record we have to disagree with the Examiner's claim interpretation. Claim 2 recites "emitting hypertext form tags" and claim 3 recites "encoding the hypertext input form."[2] Appellants' Specification on page 7, identifies that in the Hypertext Markup Language (HTML) the input form tags include "<FORM>, <INPUT>, <SELECT> and <OPTION> and can take different forms, including checkboxes, selectable lists, and textual input fields." This definition is consistent with the discussion on page 6 of Apple, which discusses HTML as including "a FORM element that provides the ability to create a fill-in form for the client . . . . [t]he user enters the information through the use of a limited number of FORM components: checkboxes;

[2] We note that in claim 3, the term "the hypertext input form" lacks antecedent basis.

radio boxes; pull-down lists; text windows and menus." Thus, we find that at the time of the invention the terms "hypertext form tags" and "hypertext input forms" were known and would be understood by one skilled in the art to be in reference to the HTML elements to create fill-in forms (to enter data). Accordingly, we construe claims 2, 3, and 4 (as claim 4 depends upon claim 2) to be limited to a method which makes use of the HTML elements (<FORM>, <INPUT>, <SELECT> and <OPTION>) to create forms for the user to fill-in (enter data).

We find that Apple teaches that the use of HTML FORM elements (<FORM>, <INPUT>, <SELECT> and <OPTION>) to create forms for a user to fill-in (enter data) is undesirable as the transmission and rendering of the web page is time consuming. (Apple 7, ll. 4-20.) Apple, teaches use of an allegedly improved method of providing data entry, by using applets. (Apple 12, ll. 21-26.) The applets are programs written in a programming language, such as Java, and are embedded into the HTML code. (Apple 13, ll. 1-8.) Thus, the applets are not HTML elements to create forms for the user to fill-in (enter data). As such, we cannot agree with the Examiner that the applets meet the limitations of claims 2 and 3.

What Applicant overlooks is that Apple teaches an alternative over the prior art process using of HTML FORM elements (<FORM>, <INPUT>, <SELECT> and <OPTION>) to create forms for a user to fill-in (enter data). Further, we are unable to find that Apple identifies that there is a restriction on the spelling of names associated with these elements. Thus, we affirm the Examiner's decision to reject of claims 2 through 4, however,

we apply different rationale than applied by the Examiner and designate the affirmance a new rejection under 37 C.F.R. § 41.50(b).[3]

Examiner's Rejection of Claims 9 and 13.

Appellants assert, on page 5 of the Brief, that dependent claims 9 and 13 recite a method that includes converting the submitted input field data to a correct data type. Appellants argue that this feature is not anticipated by Apple.

The Examiner does not address Appellants' argument directly but refers to the statement of rejection in the April 19, 2002, Final Office action. (Answer 5.) In the statement of rejection, on page 6 of the Final Office action, the Examiner refers to Apple, page 14, lines 10-13, as providing evidence that Apple anticipates converting the input field data to the correct data type.

We disagree with the Examiner as we are unable to find that the evidence supports the Examiner's finding. Apple discusses, in the paragraph on page 14, starting on line 9, mapping of HTML tags or HTML objects, including the applets, to the objects, i.e. mapping the input fields from the Hypertext document to the inputs to another application. However, we find no discussion on page 14, of converting the input data to a correct data type. Thus, we reverse the Examiner's rejection of claims 9 and 13.

[3] We note given our claim interpretation we see no difference between Appellants' claimed invention and the disclosed prior art on pages 1 through 3 of Appellants' Specification.

Examiner's Rejection of Claim 10.

Appellants assert, on page 6 of the Brief, that dependent claim 10 recites a method that includes iteratively processing input names associated with a software component property to determine if the data associated with any of the input names has been entered. Appellants argue that this feature is not taught or suggested by Apple.

The Examiner does not address Appellants' argument directly but refers to the statement of rejection in the April 19, 2002, Final Office action. (Answer 5.) In the statement of rejection, on page 10 of the Final Office action, the Examiner finds that this feature is either inherent to Apple's teachings or obvious in light of Apple's teachings. The, Examiner reasons that inherency is shown in that "the ONLY way for the CPU to detect a state change would have been to continually check for the state change." Further, the Examiner finds that Apple suggests iterative processing on page 21, lines 1-3.

We concur with the Examiner. Claim 10 recites that "the determining [from the mapping of a software component property mapped to the named input field] includes iteratively processing input names associated with a component property to determine if data associated with any of the input names has been entered." Appellants' Specification does not provide a definition of the term "iteratively." However, the common meaning of the term iteratively is "repeating or making repetition". This meaning of the term, "iteratively" is consistent with the use of the term in Appellants' Specification, which on page 9 uses the term iteratively to describe the processor as iteratively processing data fields.

Initially, we note that the Examiner has made findings regarding inherency and obviousness. Appellants' arguments on page 6 of the Brief, addresses the obviousness rationale but do not contest the Examiner's finding that in Apple the CPU continually checks for a state change. Thus, in the absence of evidence to the contrary, we accept the Examiner's finding of fact that the method taught by Apple continually checks for a state change in variable. As the state change represents a user input, and triggers a mapping, Apple teaches continuously, repeatedly checking for user input and mapping the input to the object oriented environment. Thus, we do not consider Appellants' arguments to have shown that the Examiner's finding that Apple inherently performs the claimed step of "iteratively processing input names" is in error. Appellants' arguments, on page 6 of the Brief, that Apple teaches away from iteratively processing, is similarly not convincing, as we find that Apple inherently performs the claimed step. For the forgoing reasons we affirm the Examiner's rejection of claim 10.

Examiner's Rejection of Claims 11, 15, and 17.

Appellants argue, on page 6 of the Brief, that Apple does not teach or suggest "processing [named input fields] in order of priority that are stored with the mapping of the input fields." As such Appellants argue that the subject matter of dependent claim 11 would not have been obvious over Apple. Appellants' reason:

> In order to assign priority levels in Apple, however, the system would first have to compare the values of the applet keys to the values in the dictionary, and then determine which values were different. The changed values would be assigned a higher priority than unchanged

> values. Therefore, in Apple, priority levels would have to be derived based upon the comparison of key values.

The Examiner does not address Appellants' argument directly but refers to the statement of rejection in the April 19, 2002, Final Office action. (Answer 5.) In the statement of rejection, on page 12 of the Final Office action the Examiner states:

> [A] priority of input fields is implied, because Action Coordinator 301 processes plural input fields, and because it processes "only those values that have changed since the last communication with the browser are compiled into the package." - page 29, lines 1-10. It is further implied in the observation that input field value mappings would not have been processed randomly, thus there is in inherent order. Since "priority" is generically claimed, it is believed that APPLE COMPUTER, INC implicitly meets this limitation. Otherwise, it would have been obvious to PHOSITA at the time of the invention to process input fields in a prioritized order in order to give preference to the executions of certain actions over others.

We concur with the Examiner. Claim 11, states: "wherein the determining includes processing in order of priority stored within the mapping of the input field names." Thus, claim 11 recites that there is a priority to processing input of field names and that that priority is stored with the mapping.

Even if, as Appellants assert, the priorities in Apple are impacted by a determination that the values in the dictionary have changed, using the Examiner's rationale, Apple meets the limitation of processing in a priority. One skilled in the art would recognize that Apple only operates on one value at a time, as such when more than one value changes in a synchronization cycle, some order or priority is established to process the

individual changes. Typically in computer systems instructions are performed sequentially, such that the instructions are performed in the order presented (see for example discussion in Apple page 18 line 20 through page 19 line 11, which describes operation of a HTML script as a sequence of steps being performed in order descending from the top of the script). Thus, one would expect that the order of processing would be performed following the same order used to organize the mapping. For the forgoing reasons, we affirm the Examiner's rejection of claim 10.

On page 7 of the Brief, Appellants identify limitations of claims 15 and 17, and states "[f]or at least the aforementioned reasons, claims 15 and 17 are patentable over the prior art of record." We do not consider this to be a separate argument under 37 C.F.R. 41.27 § (a)(c)(vii). Thus, we group claims 15 and 17 with claim 11, and affirm the Examiner's rejection of claims 15 and 17 for the reasons discussed *infra* with respect to claim 11.

Examiner's Rejection of Claim 14.

Appellants argue that Apple is related to a method for comparing applet key values to values in a dictionary, and teaches away from using a form and input fields. Further, Appellants argue that Apple "does not teach or suggest mapping input fields to a priority." (Brief 7.)

Appellants' arguments have not convinced us of error in the Examiner's rejection. Claim 14 is dependent upon claim 12 and recites, "wherein the name-space manager includes a table for mapping a form to input fields, and for mapping the input fields to a component property." We find no limitation in claim 14 directed to "mapping input fields to a priority"

as asserted by Appellants. Further, while claim 14 does recite mapping a "form," claim 14 differs from claims 2 and 3, discussed *supra*, in that claim 14 does not recite using "hypertext form tags" or "hypertext input form." Thus, unlike claims 2 and 3, claim 14 is not limited to a method which makes use of the HTML elements (<FORM>, <INPUT>, <SELECT> and <OPTION>) to create forms for the user to fill-in (enter data). Rather, we find that claim 14 includes mapping a form (of any type) to input fields. Apple teaches using HTML documents which permit a user to exchange information with a server. (Apple 5, ll. 21-22.) The HTML document may include elements which allow a user to fill in information. As discussed *supra* with respect to claim 2, Apple teaches that one method is to use HTML FORM elements and another is to use applets. Thus, the HTML document is a form as it allows the user to enter data, regardless of whether HTML FORM elements are used or applets. As discussed *supra* with respect to claim 1 we find that Apple teaches the claimed step of mapping input fields to software component property. For the forgoing reasons we sustain the Examiner's rejection of claim 14.

CONCLUSION

Appellants' contentions have not convinced us of error in the Examiner's rejection of claims 1 through 4, 10 through 12, and 14 through 17 and we affirm the rejection of these claims. However, Appellants have convinced us of error in the Examiner's rejection of claims 9 and 13 and reverse this rejection.

ORDER

The decision of the Examiner is affirmed-in-part.

Regarding the affirmed rejection(s), 37 C.F.R. § 41.52(a)(1) provides "[a]ppellant may file a single request for rehearing within two months from the date of the original decision of the Board."

In addition to affirming the examiner's rejection(s) of one or more claims, this decision contains a new ground of rejection pursuant to 37 C.F.R. § 41.50(b) (effective September 13, 2004, 69 Fed. Reg. 49960 (August 12, 2004), 1286 Off. Gaz. Pat. Office 21 (September 7, 2004)). 37 C.F.R. § 41.50(b) provides "[a] new ground of rejection pursuant to this paragraph shall not be considered final for judicial review."

37 C.F.R. § 41.50(b) also provides that the appellant, WITHIN TWO MONTHS FROM THE DATE OF THE DECISION, must exercise one of the following two options with respect to the new ground of rejection to avoid termination of the appeal as to the rejected claims:

> (1) *Reopen prosecution.* Submit an appropriate amendment of the claims so rejected or new evidence relating to the claims so rejected, or both, and have the matter reconsidered by the examiner, in which event the proceeding will be remanded to the examiner. . . .
>
> (2) *Request rehearing.* Request that the proceeding be reheard under § 41.52 by the Board upon the same record. . . .

Should the appellant elect to prosecute further before the examiner pursuant to 37 C.F.R. § 41.50(b)(1), in order to preserve the right to seek review under 35 U.S.C. §§ 141 or 145 with respect to the affirmed rejection,

the effective date of the affirmance is deferred until conclusion of the prosecution before the examiner unless, as a mere incident to the limited prosecution, the affirmed rejection is overcome.

If the appellant elects prosecution before the examiner and this does not result in allowance of the application, abandonment or a second appeal, this case should be returned to the Board of Patent Appeals and Interferences for final action on the affirmed rejection, including any timely request for rehearing thereof.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a). *See* 37 C.F.R. § 1.136(a)(1)(iv).

AFFIRMED-IN-PART
37 C.F.R. § 41.50(b)

rwk

MICHAEL A DIENER
HALE AND DORR
60 STATE STREET
BOSTON MA 02109